Java Programming and Design Lecture 2 – Review, IntelliJ and Parameters

Chris Loftus, B62

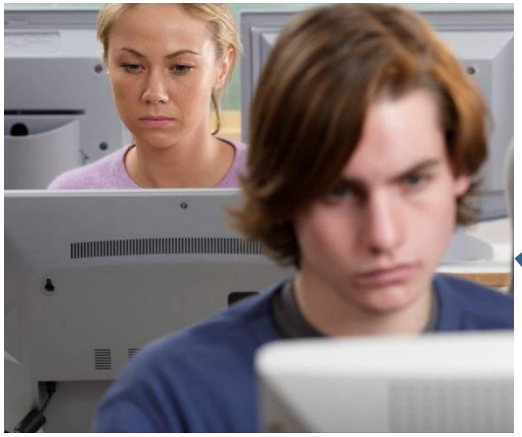
cwl@aber.ac.uk

Qwizdom in lectures

- We only have a 100 seat license so please pair up an one person use smartphone etc
  - qvr.qwizdon.com Q5VN94
- When question asked discuss as a pair and then select an answer
- We will then discuss

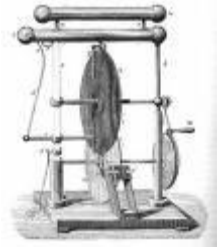
Remember our Overview of Java

Translated/
compiled



JAVA Program

Byte Code Program



write

You

Java Virtual Machine runs it –
things happen

Quiz: Why do we often use a compiler? (Choose all that apply)

A) To check for errors in our code before we run it

B) To run our code

C) To translate our code to a more efficient form

D) Not sure

Quiz: Why is the Java Virtual Machine Virtual?

(Choose most accurate answer)

A) It simulates a real computer so that our byte-code runs everywhere

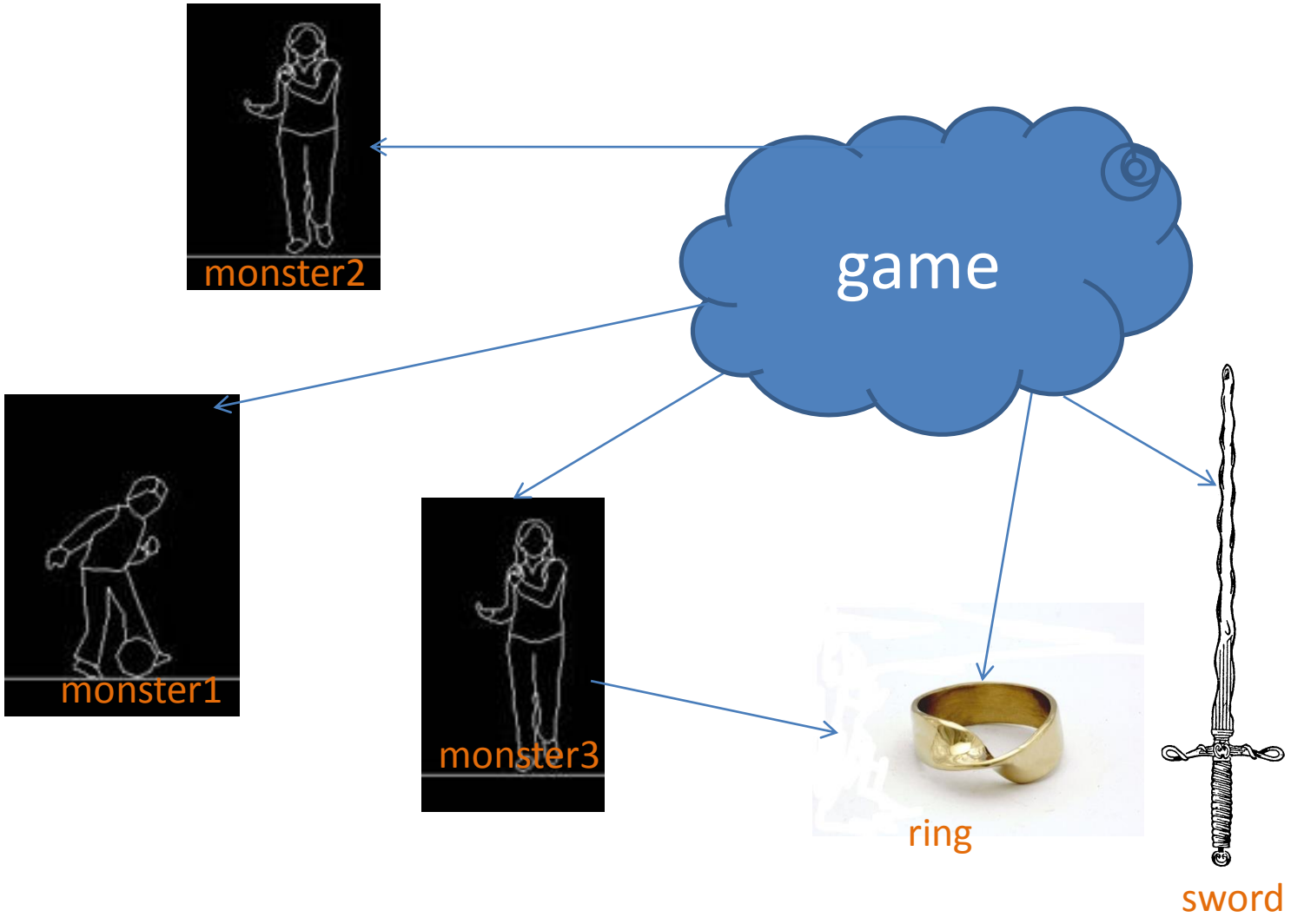
B) Because it doesn't really exist

C) Not sure

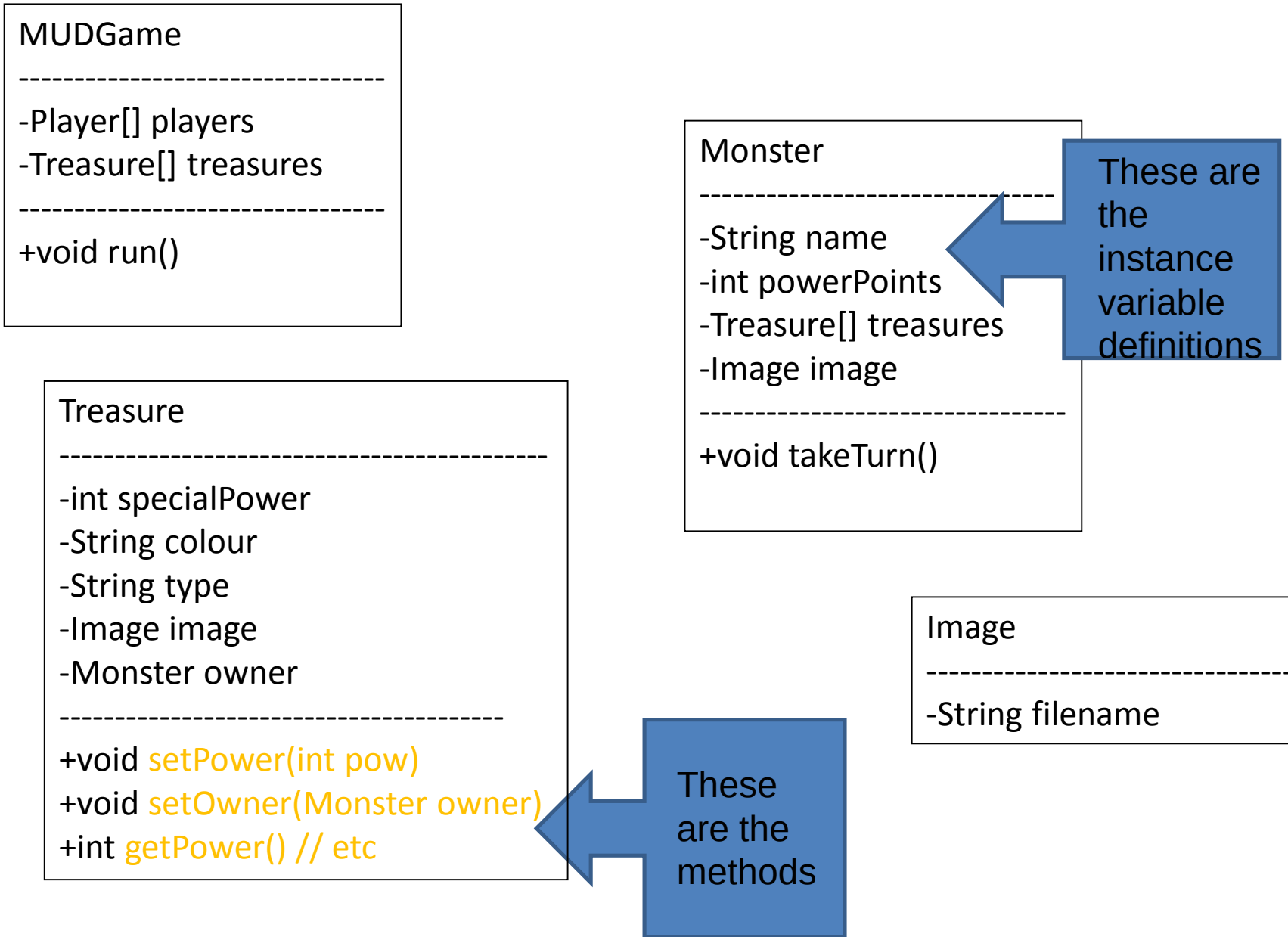
Quiz: How is an interpreted language different from a compiled language? (Choose all that apply)

A) It is usually more efficient and runs faster
B) It is usually less efficient and runs slower
C) It allows for more errors to occur at runtime
D) Interpreted programs are quicker to change ready for running
E) Not sure

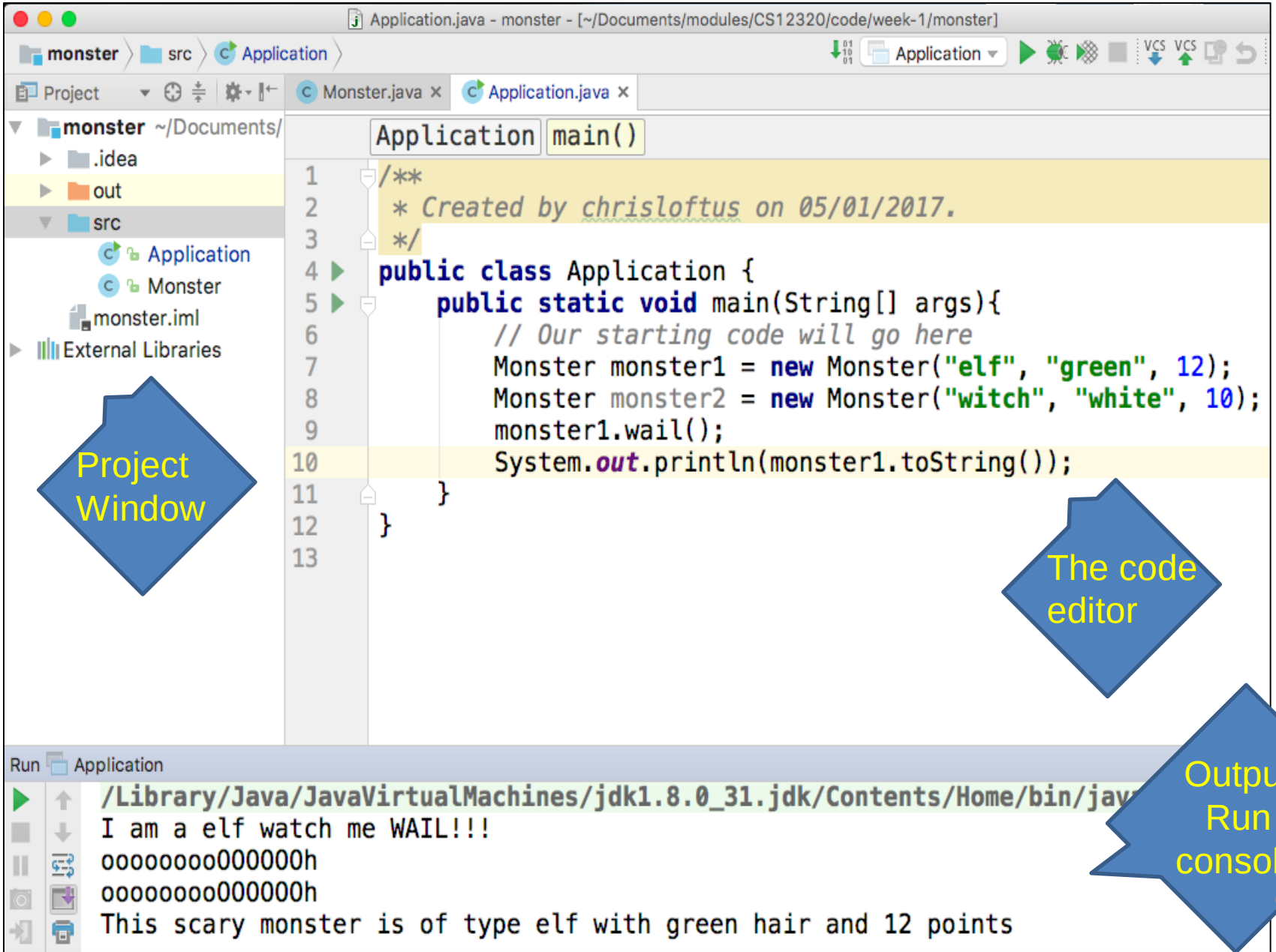
Our object diagrams (as the program runs)



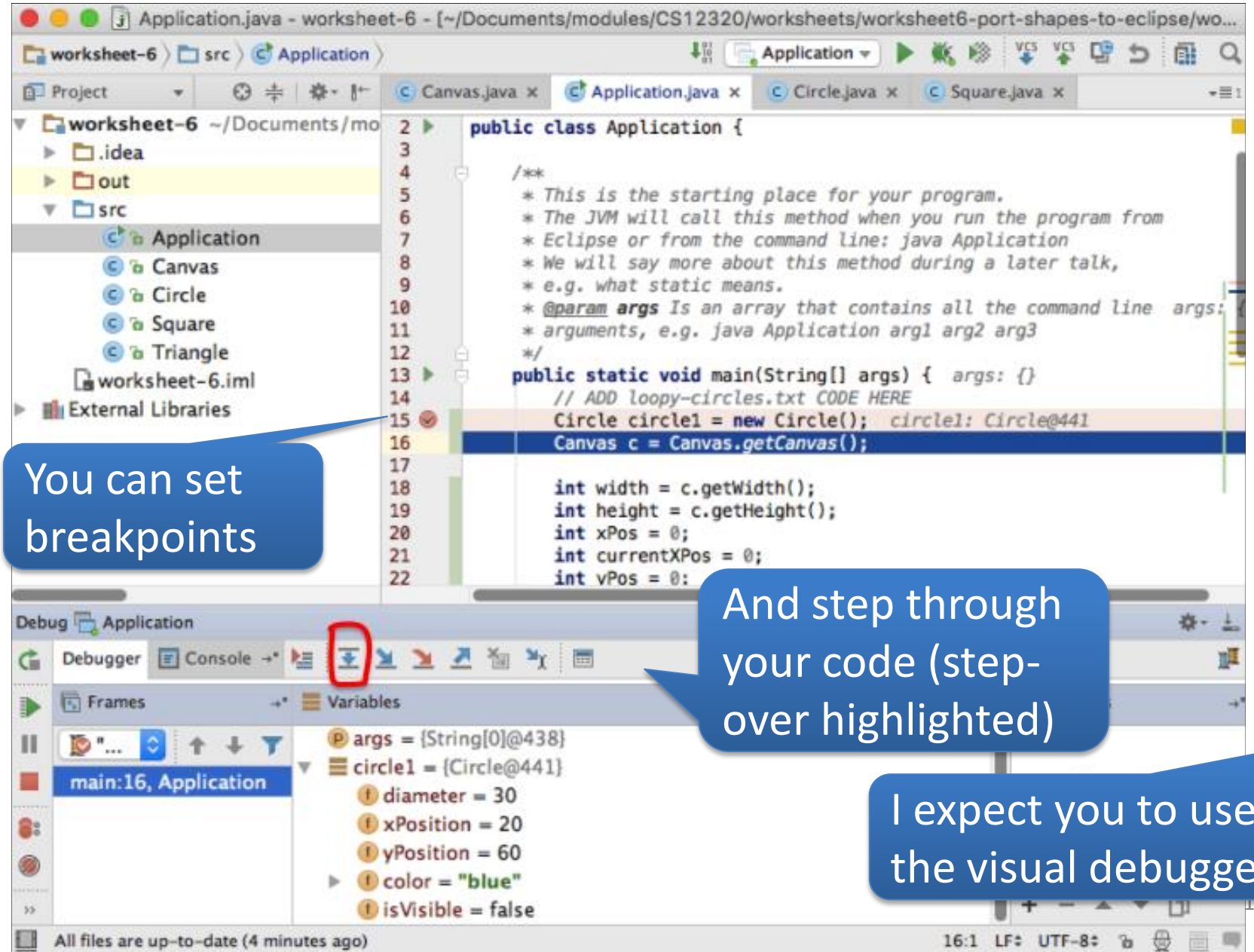
and our Class diagram (for design)



So what did IntelliJ give us?

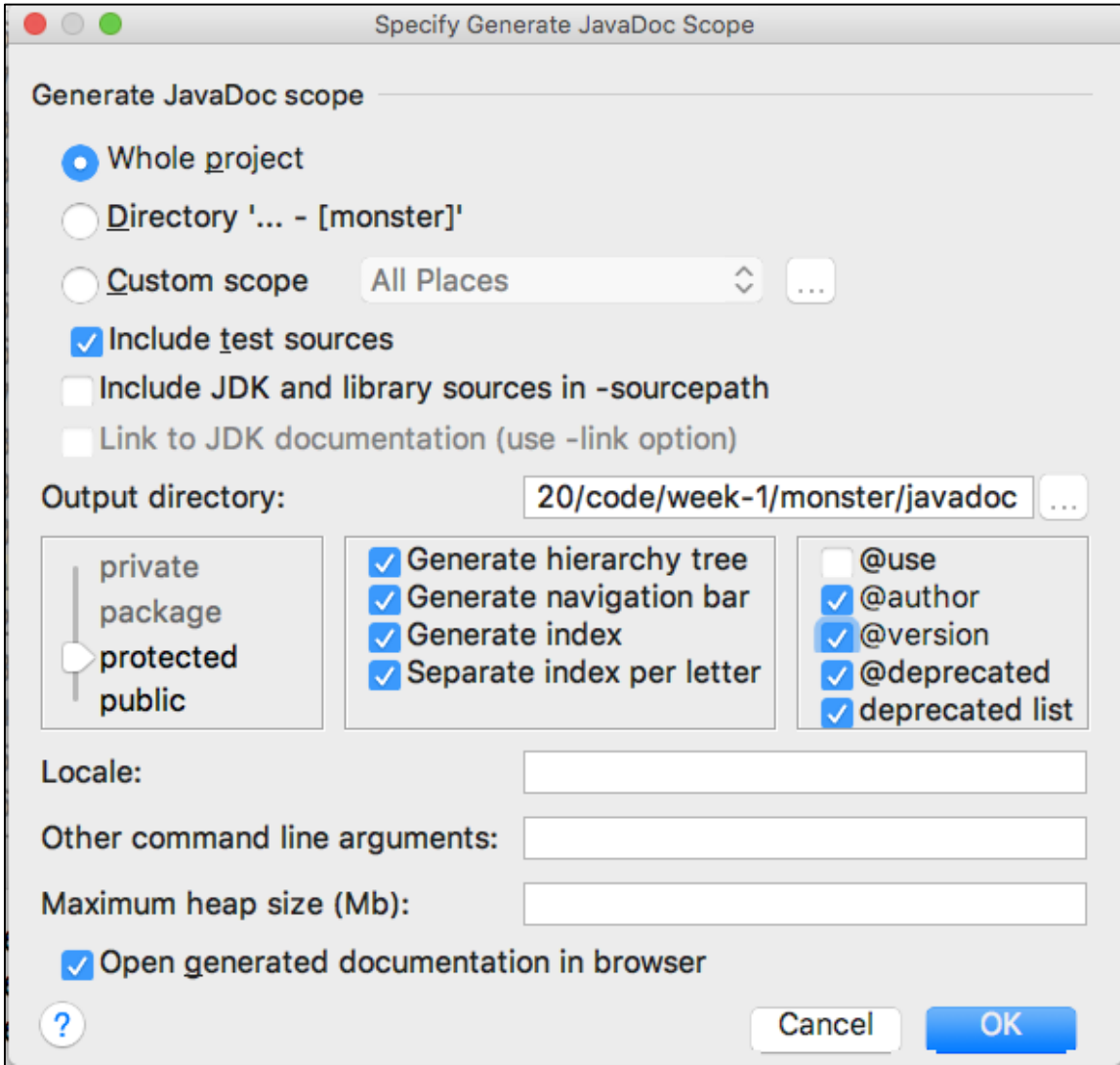


Also visual debugging



And generating Java API documentation

- Generated from JavaDoc comments (next slide)
- Tools->Generate JavaDoc
- Same style as official Oracle JavaDoc API http://docs.oracle.com/javase/8/docs/api/



```java
/**
 * This is the class Monster - a 'blueprint' for all Monsters
 *
 * @author Chris Loftus
 * @version 1, 6th January 2017
 */
public class Monster {

    /**
     * Constructor for objects of class Monster
     */
    public Monster() {...}

    /**
     * Constructor for objects of class Monster
     *
     * @param startType   becomes initial type of monster
     * @param startHair   becomes initial hair colour
     * @param startPoints initial power points
     */
    public Monster(String startType, String startHair, int startPoints) {...}

    /**
     * This method 'returns' information about a monster
     *
     * @return monster info for printing
     */
    public String toString() {...}

    // Other code omitted
}
```

JavaDoc comments start with /**

Always provide a header JavaDoc comment

Every non-private constructor / method should have one

@param for a parameter and @return for a return value

All Classes

Application
Monster

PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Monster

java.lang.Object
    Monster

```
public class Monster
extends java.lang.Object
```

This is the class Monster - a 'blueprint' for all Monsters

Version:

`1, 6th January 2017`

Author:

`Chris Loftus`

Constructor Summary

Constructors

Constructor and Description
`Monster()` Constructor for objects of class Monster
`Monster(java.lang.String startType, java.lang.String startHair, int startPoints)` Constructor for objects of class Monster

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`java.lang.String`	**`getHair()`** This method gets the hair colour

Monster

```
public Monster(java.lang.String startType,
               java.lang.String startHair,
               int startPoints)
```

Constructor for objects of class Monster

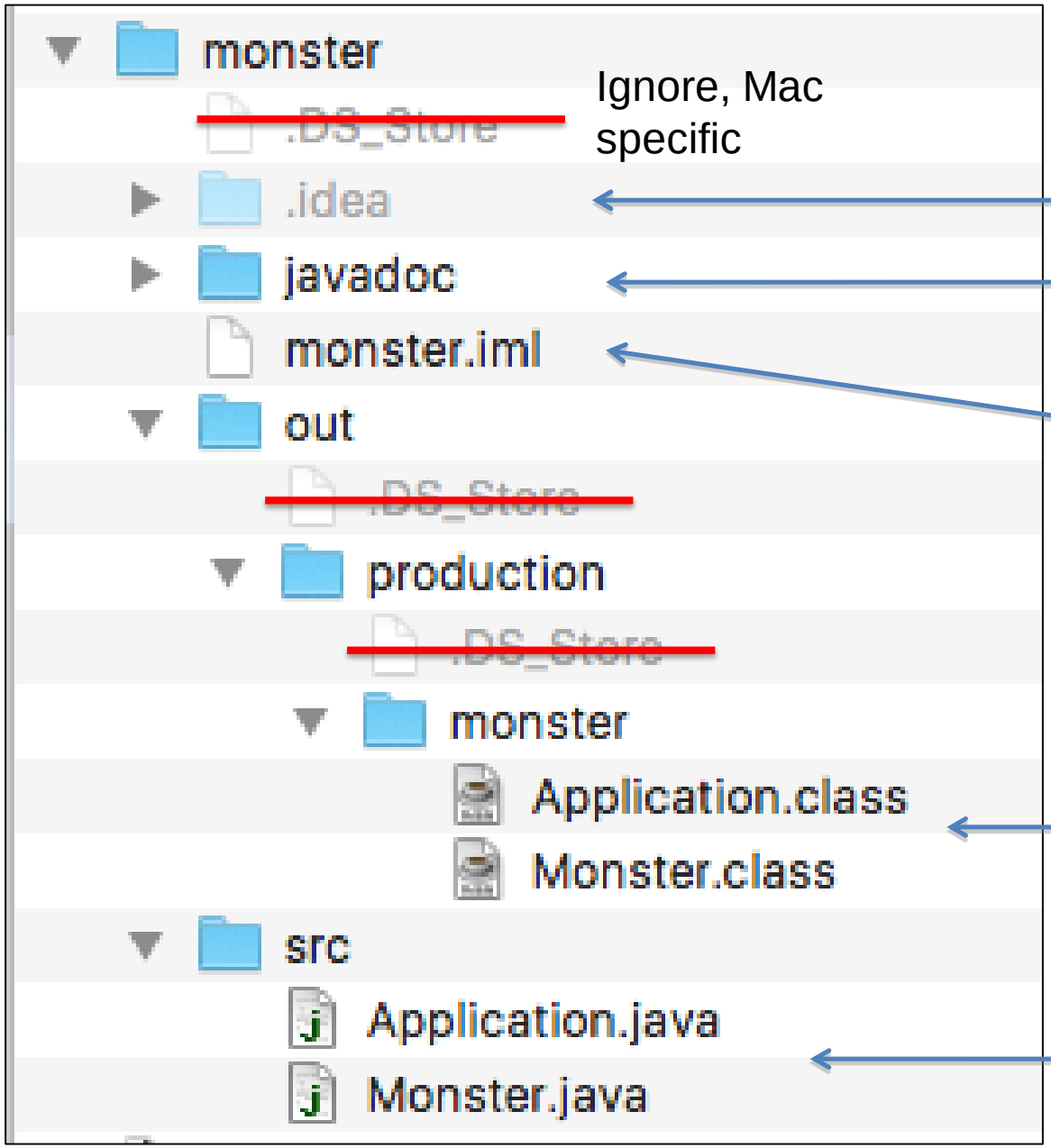
Parameters:

`startType` – becomes initial type of monster

`startHair` – becomes initial hair colour

`startPoints` – initial power points

What files did IntelliJ give you?



Ignore, Mac specific

Contains project settings

Any generated Java documentation

Describes the project, e.g. where the source is found

Java class bytecode is here

Java source code is under src folder

Things I didn't tell you

- Comments:
  - everything between /* …… */
  - everything between /** ……*/
  - or rest of line from //
- Variable/method names mustn't have spaces and should start with lower case e.g. specialPower (uppercase for other words)
- Class names should start with upper case e.g. HelloWorld
- Return from methods…
  - public String toString() vs. public void setName(String n)

What “methods” should you always have?

- Constructor — public Person() {…
- toString() — public String toString() {…
- Gets — public String getName() {…
- Sets — public void setName (String n) {…

There are a couple more (e.g. equals) that we’ll come back to…

But in order to do anything ‘interesting’ we need more

More on parameters

How you get a value into a method/constructor to change instance variables?

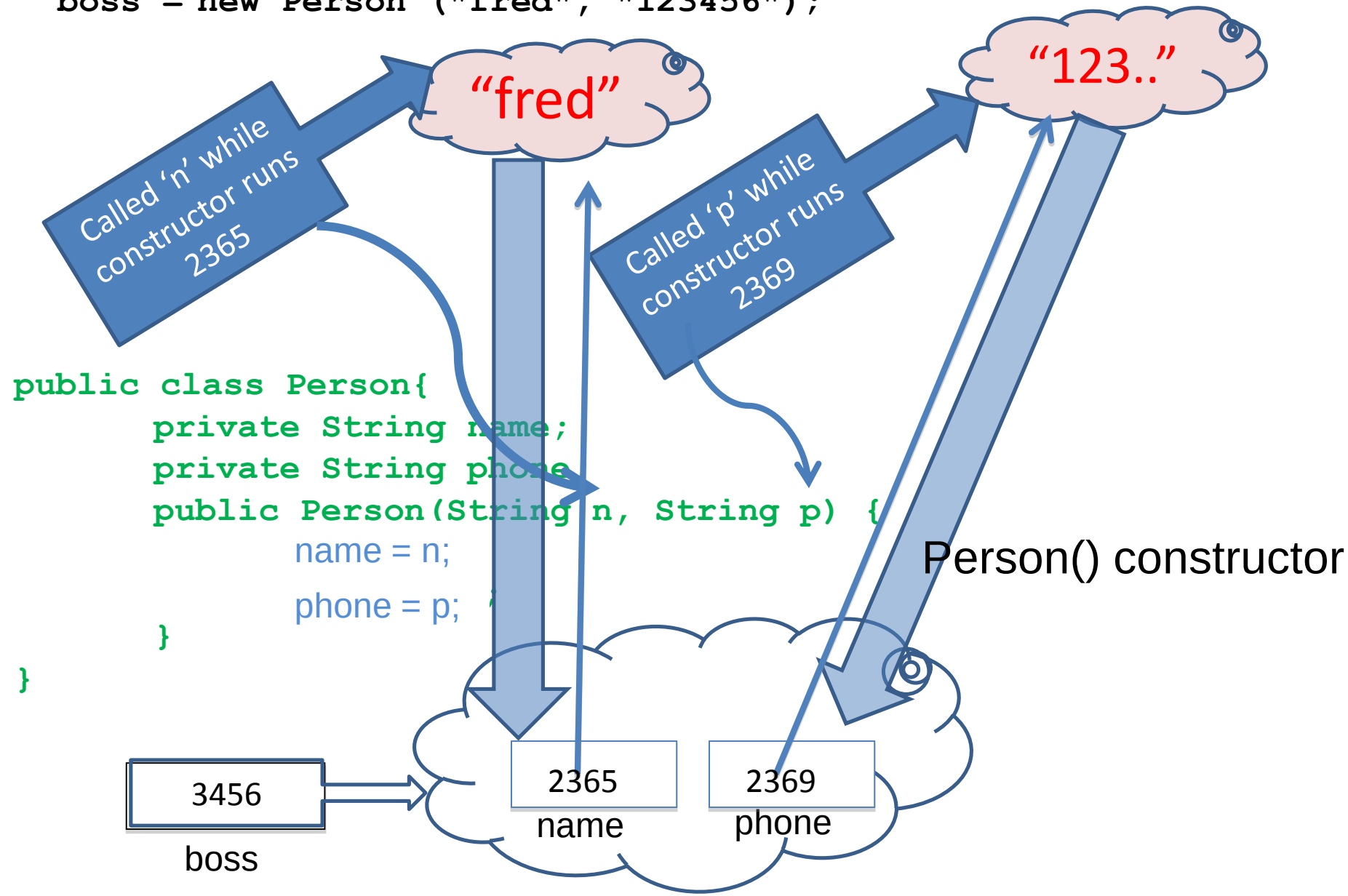
```
public class Person{
   private String name;
   private String phone;
   public Person(String n, String p) {
       name = n;
       phone = p;
   }
}
```

Call this as:

```
Person boss = new Person("fred", "123456");
```

What happens?

```
Person boss;
boss = new Person ("fred", "123456");
```



We have done a lot!

- What is a class?
- What is an object?
- Design a class
- Looked at the code for it
- Learned to use IntelliJ basics

Workshops this week

Workshop 3

- Some more qwizdom
- **Small, assessed in-class exercise** on finding classes and links. Similar to last Monday's lecture in-class exercise: signed off by demonstrator and uploaded to BB
- Worksheet 7: Continue

Workshop 4

- Talk on relationships and UML class diagrams
- **Final opportunity to sign-off worksheets 6 + 7**
- Worksheet 8